R Programming – Lab File

by

Danveer

102203272

3CO7

Submitted to

Dr. Rajneesh Kumar Rai

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

DEPARTMENT OF MATHEMATICS

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY, (ADEEMED TO BE UNIVERSITY), PATIALA, PUNJAB

INDIA

Session-Year (e.g. Jul-Dec, 2024)

Probability and Statistics (UMA401)
Experiment 1: Basics of R programming

(1) Create a vector c = [5,10,15,20,25,30] and write a program which returns the maximum and minimum of this vector.
(2) Write a program in R to find factorial of a number by taking input from user. Please print error message if the input number is negative.
(3) Write a program to write first n terms of a Fibonacci sequence. You may take n as an input from the user.
(4) Write an R program to make a simple calculator which can add, subtract, multiply and divide.
(5) Explore plot, pie, barplot etc. (the plotting options) which are built-in functions in R.

Solution:

```r
#Question 1
#Create a vector c = [5,10,15,20,25,30] and write a program which returns the maximum and minimum of this vector.

vec <- c(5,10,15,20,25,30)

max_val <- max(vec)
min_val <- min(vec)

cat("Maximum Value", max_val, "\n")
cat("Minimum Value", min_val, "\n")


#Question 2
#Write a program in R to find factorial of a number by taking input from user. Please
#print error message if the input number is negative.

factorial <- function(n) {
  if(n < 0) {
    return ("Error: Factorial of negative number doesn't exist.")
  } else if(n == 0) {
    return (1)
  } else {
    return (n * factorial(n-1))
  }
}

num <- as.integer(readline(prompt = "Enter number: "))
cat(num, "! = ", factorial(num))


#Question 3
#Write a program to write first n terms of a Fibonacci sequence. You may take n as an
#input from the user.
```

```r
fibonacci <- function(n) {
  fib <- numeric(n)
  fib[1] <- 0
  if(n > 1) {
    fib[2] <- 1

    for(i in 3:n) {
      fib[i] <- fib[i-1] + fib[i-2]
    }
  }

  return(fib)
}

n <- as.integer(readline(prompt="Enter the number of terms: "))

cat("First", n, "terms of Fibonacci sequence are:\n")
print(fibonacci(n))



#Question 4
#Write an R program to make a simple calculator which can add, subtract, multiply
#and divide.

add <- function(a, b) {
  return(a + b)
}

subtract <- function(a, b) {
  return(a - b)
}

multiply <- function(a, b) {
  return(a * b)
}

divide <- function(a, b) {
  if(b == 0) {
    return("Error: Division by zero is not allowed.")
  } else {
    return(a / b)
  }
}

a <- as.numeric(readline(prompt="Enter first number: "))
b <- as.numeric(readline(prompt="Enter second number: "))
operation <- readline(prompt="Enter operation (+, -, *, /): ")
```

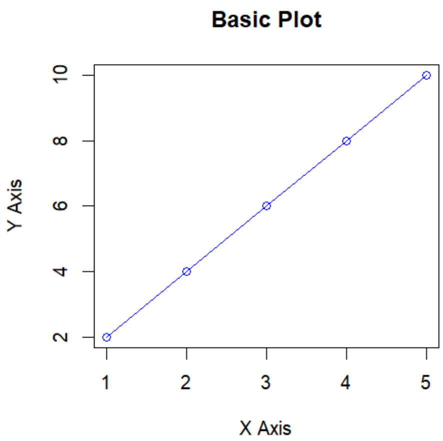
```r
if(operation == "+") {
  cat("Result:", add(a, b), "\n")
} else if(operation == "-") {
  cat("Result:", subtract(a, b), "\n")
} else if(operation == "*") {
  cat("Result:", multiply(a, b), "\n")
} else if(operation == "/") {
  cat("Result:", divide(a, b), "\n")
} else {
  cat("Invalid operation.\n")
}



#Question 5
#Explore plot, pie, barplot etc. (the plotting options) which are built-in functions in R.
# Sample data for plotting
x <- c(1, 2, 3, 4, 5)
y <- c(2, 4, 6, 8, 10)

# Basic plot
plot(x, y, type="o", col="blue", main="Basic Plot", xlab="X Axis", ylab="Y Axis")

# Pie chart
slices <- c(10, 20, 30, 40)
labels <- c("A", "B", "C", "D")
pie(slices, labels=labels, main="Pie Chart", col=rainbow(length(slices)))

# Bar plot
barplot(slices, names.arg=labels, col=rainbow(length(slices)), main="Bar Plot",
xlab="Categories", ylab="Values")

# Histogram
data <- rnorm(100)
hist(data, col="lightblue", main="Histogram", xlab="Values", ylab="Frequency")
```

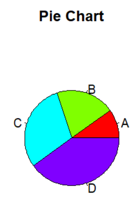
Output:

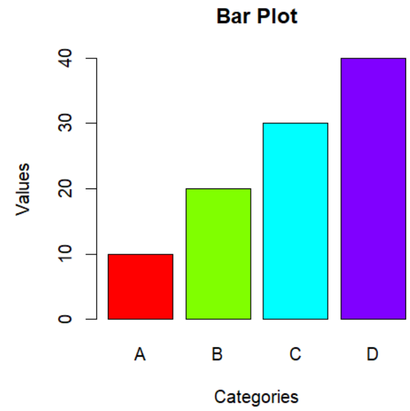
```
> #Question 1
> #Create a vector c = [5,10,15,20,25,30] and write a program which returns the maximum and minimum of this vector.
> print("Question 1")
[1] "Question 1"
> vec <- c(5,10,15,20,25,30)
> max_val <- max(vec)
> min_val <- min(vec)
> cat("Maximum Value", max_val, "\n")
Maximum Value 30
> cat("Minimum Value", min_val, "\n")
Minimum Value 5
> print("--------------------------------------------------------------------------------")
[1] "--------------------------------------------------------------------------------"
> print("Question 2")
[1] "Question 2"
> factorial <- function(n) {
+   if(n < 0) {
+     return ("Error: Factorial of negative number doesn't exist.")
+   } else if(n == 0) {
+     return (1)
+   } else {
+     return (n * factorial(n-1))
+   }
+ }
> num <- as.integer(readline(prompt = "Enter number: "))
Enter number: 3
> cat(num, "! = ", factorial(num))
3 ! =  6
> print("--------------------------------------------------------------------------------")
[1] "--------------------------------------------------------------------------------"
> #Question 3
> #Write a program to write first n terms of a Fibonacci sequence. You may take n as an
> #input from the user.
> print("Question 3")
[1] "Question 3"
> fibonacci <- function(n) {
+   fib <- numeric(n)
+   fib[1] <- 0
+   if(n > 1) {
+     fib[2] <- 1
+
+     for(i in 3:n) {
```

```
> cat("First", n, "terms of Fibonacci sequence are:\n")
First 5 terms of Fibonacci sequence are:
> print(fibonacci(n))
[1] 0 1 1 2 3
> print("--------------------------------------------------------------------------------")
[1] "--------------------------------------------------------------------------------"
> #Question 4
> #Write an R program to make a simple calculator which can add, subtract, multiply
> #and divide.
> print("Question 4")
[1] "Question 4"
> add <- function(a, b) {
+   return(a + b)
+ }
> subtract <- function(a, b) {
+   return(a - b)
+ }
> multiply <- function(a, b) {
+   return(a * b)
+ }
> divide <- function(a, b) {
+   if(b == 0) {
+     return("Error: Division by zero is not allowed.")
+   } else {
+     return(a / b)
+   }
+ }
> a <- as.numeric(readline(prompt="Enter first number: "))
Enter first number: 6
> b <- as.numeric(readline(prompt="Enter second number: "))
Enter second number: 2
> operation <- readline(prompt="Enter operation (+, -, *, /): ")
Enter operation (+, -, *, /): +
> if(operation == "+") {
+   cat("Result:", add(a, b), "\n")
+ } else if(operation == "-") {
+   cat("Result:", subtract(a, b), "\n")
+ } else if(operation == "*") {
+   cat("Result:", multiply(a, b), "\n")
+ } else if(operation == "/") {
+   cat("Result:", divide(a, b), "\n")
+ } else {
```

```
+ }
Result: 8
> print("--------------------------------------------------------------------------------")
[1] "--------------------------------------------------------------------------------"
> #Question 5
> #Explore plot, pie, barplot etc. (the plotting options) which are built-in functions in R.
> # Sample data for plotting
> print("Question 5")
[1] "Question 5"
> x <- c(1, 2, 3, 4, 5)
> y <- c(2, 4, 6, 8, 10)
> # Basic plot
> plot(x, y, type="o", col="blue", main="Basic Plot", xlab="X Axis", ylab="Y Axis")
> # Pie chart
> slices <- c(10, 20, 30, 40)
> labels <- c("A", "B", "C", "D")
> pie(slices, labels=labels, main="Pie Chart", col=rainbow(length(slices)))
> # Bar plot
> barplot(slices, names.arg=labels, col=rainbow(length(slices)), main="Bar Plot", xlab="Categories", ylab="Values")
> # Histogram
> data <- rnorm(100)
> hist(data, col="lightblue", main="Histogram", xlab="Values", ylab="Frequency")
> print("--------------------------------------------------------------------------------")
[1] "--------------------------------------------------------------------------------"
>
```







Probability and Statistics (UMA401)
Experiment 2: Descriptive statistics, Sample space, definition of Probability

```
# (1) (a) Suppose there is a chest of coins with 20 gold, 30 silver and 50 bronze coins. You
#     randomly draw 10 coins from this chest. Write an R code which will give us the
#     sample space for this experiment. (use of sample(): an in-built function in R)
#     (b) In a surgical procedure, the chances of success and failure are 90% and 10%
#     respectively. Generate a sample space for the next 10 surgical procedures performed.
#     (use of prob(): an in-built function in R)

#Part (a)
#Creating a vector containing 20 gold, 30 silver and 50 bronze
# Note: rep --> This is a function for returning a vector which repeat the value n times

treasure_chest <- c(rep('Gold', 20), rep('Silver', 30), rep('Bronze', 50))
sample_space_chest <- sample(x = treasure_chest, size = 10, replace = TRUE)
cat("Sample Space of Treasure Chest: ", sample_space_chest, "\n")
#Here replace is true because repetition is allowed

#Part (b)
#Note: Every next procedure can be either success or failure having prob --> 90%, 10%
#Note: Every procedure is independent of each other i.e. Repetition is allowed
sample_space_procedure <- sample(x = c('Success', 'Failure (Honsla Rakho)'), size = 10,
replace = TRUE, prob = c(0.9, 0.1))
cat("Sample Space of Procedures: ", sample_space_procedure, "\n")



# (2) A room has n people, and each has an equal chance of being born on any of the 365
# days of the year. (For simplicity, we'll ignore leap years). What is the probability
```

```r
# that two people in the room have the same birthday?
#   (a) Use an R simulation to estimate this for various n.
#   (b) Find the smallest value of n for which the probability of a match is greater than
0.5.

#Part (a)
n <- 50
prob_no_shared_birthday <- 1 #Taking initial probability (365/365), This is first term

#Note probability in each iteration --> ((365 - (n-1))/365)
for(i in 2:(n)) {
  prob_no_shared_birthday <- prob_no_shared_birthday * ((365 - (i-1))/ 365)
}

prob_shared_birthday <- (1-prob_no_shared_birthday)
cat("Probabilty of two people having same birthday: ", prob_shared_birthday, "\n")

#Part(b)
initial_people <- 1;
prob_no_shared_birthday <-1 #Let say initially prob of no shared birthday is 1

find_min_people <- function(current_people, prob_no_shared_birthday) {
  for(i in 2:current_people) {
    prob_no_shared_birthday <- prob_no_shared_birthday * ((365 - (i-1))/365)
  }

  prob_shared_birthday <- 1 - prob_no_shared_birthday

  if(prob_shared_birthday > 0.5) {
    cat("Probability of Shared Birthday: ", prob_shared_birthday, "\n")
    cat("Minimum people: ", current_people, "\n")
  }

  else {
    current_people <- current_people + 1
    find_min_people(current_people , 1)
  }
}

find_min_people(initial_people, prob_no_shared_birthday)




# (3) Write an R function for computing conditional probability. Call this function to do
# the following problem:
#
#   Suppose the probability of the weather being cloudy is 40%. Also suppose the prob-
#   ability of rain on a given day is 20% and that the probability of clouds on a rainy day
#   is 85%. If it’s cloudy outside on a given day, what is the probability that it will rain
```

```r
#   that day?

#Creating function
prob_rain_given_cloudy <- function(pR, pCR, pC) {
  #Note: Here pR --> prob of Rain
              # pCR --> prob of Cloudy given Rain
              # pC --> prob of Cloudy
  #By Bayes Theorem
  pRC <- (pR * pCR)/pC
  return (pRC)
}

#Given values
pR <- 0.2
pCR <- 0.85
pC <- 0.4

cat("Probablity of Rain given Cloudy: ", prob_rain_given_cloudy(pR, pCR, pC), "\n")




# (4) The iris dataset is a built-in dataset in R that contains measurements on 4 different
# attributes (in centimeters) for 150 flowers from 3 different species. Load this dataset
# and do the following:
    data("iris")
#   (a) Print first few rows of this dataset.
    head(iris)
#   (b) Find the structure of this dataset.
    str(iris)
#   (c) Find the range of the data regarding the sepal length of flowers.
    range(iris$Sepal.Length)
#   (d) Find the mean of the sepal length.
    mean(iris$Sepal.Length)
#   (e) Find the median of the sepal length.
    median(iris$Sepal.Length)
#   (f) Find the first and the third quartiles and hence the interquartile range.
    quantile(iris$Sepal.Length, c(0.25, 0.75))
    IQR(iris$Sepal.Length)
#   (g) Find the standard deviation and variance.
    sd(iris$Sepal.Length)
    var(iris$Sepal.Length)
#   (h) Try doing the above exercises for sepal.width, petal.length and petal.width.
    # Sepal.Width
    mean(iris$Sepal.Width)
    median(iris$Sepal.Width)
    quantile(iris$Sepal.Width, c(0.25, 0.75))
    IQR(iris$Sepal.Width)
    sd(iris$Sepal.Width)
    var(iris$Sepal.Width)
```

```r
    # Petal.Length
    mean(iris$Petal.Length)
    median(iris$Petal.Length)
    quantile(iris$Petal.Length, c(0.25, 0.75))
    IQR(iris$Petal.Length)
    sd(iris$Petal.Length)
    var(iris$Petal.Length)

    # Petal.Width
    mean(iris$Petal.Width)
    median(iris$Petal.Width)
    quantile(iris$Petal.Width, c(0.25, 0.75))
    IQR(iris$Petal.Width)
    sd(iris$Petal.Width)
    var(iris$Petal.Width)

#   (i) Use the built-in function summary on the dataset Iris.
    summary(iris)





# (5) R does not have a standard in-built function to calculate mode. So we create a user
# function to calculate mode of a data set in R. This function takes the vector as input
# and gives the mode value as output.

find_mode <- function(vector) {
  #Creating frequency table of vector
  freq_table <- table(vector)

  #Find the mode value with highest frequency
  mode_value <- as.numeric(names(freq_table)[which.max(freq_table)])

  return (mode_value)
}

#Initializing vector
vector <- c(2, 3, 3, 5, 7, 7, 7, 8, 8, 10)
mode_value <- find_mode(vector)
cat("Mode: ", mode_value, "\n")
```

Output:

```
> treasure_chest <- c(rep('Gold', 20), rep('Silver', 30), rep('Bronze', 50))
> sample_space_chest <- sample(x = treasure_chest, size = 10, replace = TRUE)
> cat("Sample Space of Treasure Chest: ", sample_space_chest, "\n")
Sample Space of Treasure Chest:  Bronze Silver Bronze Silver Bronze Bronze Gold Bronze Gold Bronze
> #Part (b)
> #Note: Every next procedure can be either success or failure having prob --> 90%, 10%
> #Note: Every procedure is independent of each other i.e. Repetition is allowed
> sample_space_procedure <- sample(x = c('Success', 'Failure (Honsla Rakho)'), size = 10, replace = TRUE, prob = c(0.9, 0.1))
> cat("Sample Space of Procedures: ", sample_space_procedure, "\n")
Sample Space of Procedures:  Success Success Success Failure (Honsla Rakho) Success Success Success Success Success Success
> #Part (a)
> n <- 50
> prob_no_shared_birthday <- 1 #Taking initial probability (365/365), This is first term
> #Note probability in each iteration --> ((365 - (n-1))/365)
> for(i in 2:(n)) {
+   prob_no_shared_birthday <- prob_no_shared_birthday * ((365 - (i-1))/ 365)
+ }
> prob_shared_birthday <- (1-prob_no_shared_birthday)
> cat("Probabilty of two people having same birthday: ", prob_shared_birthday, "\n")
Probabilty of two people having same birthday:  0.9703736
> #Part(b)
> initial_people <- 1;
> prob_no_shared_birthday <-1 #Let say initially prob of no shared birthday is 1
> find_min_people <- function(current_people, prob_no_shared_birthday) {
+   for(i in 2:current_people) {
+     prob_no_shared_birthday <- prob_no_shared_birthday * ((365 - (i-1))/365)
+   }
+
+   prob_shared_birthday <- 1 - prob_no_shared_birthday
+
+   if(prob_shared_birthday > 0.5) {
+     cat("Probability of Shared Birthday: ", prob_shared_birthday, "\n")
+     cat("Minimum people: ", current_people, "\n")
+   }
+
+   else {
+     current_people <- current_people + 1
+     find_min_people(current_people , 1)
+   }
+ }
> find_min_people(initial_people, prob_no_shared_birthday)
Probability of Shared Birthday:  0.5072972
```

```
> find_min_people(initial_people, prob_no_shared_birthday)
Probability of Shared Birthday:  0.5072972
Minimum people:  23
> #Creating function
> prob_rain_given_cloudy <- function(pR, pCR, pC) {
+   #Note: Here pR --> prob of Rain
+               # pCR --> prob of Cloudy given Rain
+               # pC --> prob of Cloudy
+   #By Bayes Theorem
+   pRC <- (pR * pCR)/pC
+   return (pRC)
+ }
> #Given values
> pR <- 0.2
> pCR <- 0.85
> pC <- 0.4
> cat("Probablity of Rain given Cloudy: ", prob_rain_given_cloudy(pR, pCR, pC), "\n")
Probablity of Rain given Cloudy:  0.425
> # (4) The iris dataset is a built-in dataset in R that contains measurements on 4 different
> # attributes (in centimeters) for 150 flowers from 3 different species. Load this dataset
> # and do the following:
>     data("iris")
> #   (a) Print first few rows of this dataset.
>     head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> #   (b) Find the structure of this dataset.
>     str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> #   (c) Find the range of the data regarding the sepal length of flowers.
>     range(iris$Sepal.Length)
[1] 4.3 7.9
```

```
> #   (d) Find the mean of the sepal length.
>     mean(iris$Sepal.Length)
[1] 5.843333
> #   (e) Find the median of the sepal length.
>     median(iris$Sepal.Length)
[1] 5.8
> #   (f) Find the first and the third quartiles and hence the interquartile range.
>     quantile(iris$Sepal.Length, c(0.25, 0.75))
25% 75%
5.1 6.4
>     IQR(iris$Sepal.Length)
[1] 1.3
> #   (g) Find the standard deviation and variance.
>     sd(iris$Sepal.Length)
[1] 0.8280661
>     var(iris$Sepal.Length)
[1] 0.6856935
> #   (h) Try doing the above exercises for sepal.width, petal.length and petal.width.
>     # Sepal.Width
>     mean(iris$Sepal.Width)
[1] 3.057333
>     median(iris$Sepal.Width)
[1] 3
>     quantile(iris$Sepal.Width, c(0.25, 0.75))
25% 75%
2.8 3.3
>     IQR(iris$Sepal.Width)
[1] 0.5
>     sd(iris$Sepal.Width)
[1] 0.4358663
>     var(iris$Sepal.Width)
[1] 0.1899794
>     # Petal.Length
>     mean(iris$Petal.Length)
[1] 3.758
>     median(iris$Petal.Length)
[1] 4.35
>     quantile(iris$Petal.Length, c(0.25, 0.75))
25% 75%
1.6 5.1
>     IQR(iris$Petal.Length)
[1] 3.5
```

```
>     sd(iris$Petal.Length)
[1] 1.765298
>     var(iris$Petal.Length)
[1] 3.116278
>     # Petal.Width
>     mean(iris$Petal.Width)
[1] 1.199333
>     median(iris$Petal.Width)
[1] 1.3
>     quantile(iris$Petal.Width, c(0.25, 0.75))
25% 75%
0.3 1.8
>     IQR(iris$Petal.Width)
[1] 1.5
>     sd(iris$Petal.Width)
[1] 0.7622377
>     var(iris$Petal.Width)
[1] 0.5810063
> #   (i) Use the built-in function summary on the dataset Iris.
>     summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
> find_mode <- function(vector) {
+   #Creating frequency table of vector
+   freq_table <- table(vector)
+
+   #Find the mode value with highest frequency
+   mode_value <- as.numeric(names(freq_table)[which.max(freq_table)])
+
+   return (mode_value)
+ }
> #Initializing vector
> vector <- c(2, 3, 3, 5, 7, 7, 7, 8, 8, 10)
> mode_value <- find_mode(vector)
> cat("Mode: ", mode_value, "\n")
Mode:  7
```

Probability and Statistics (UMA401)
Experiment 3: Probability distributions

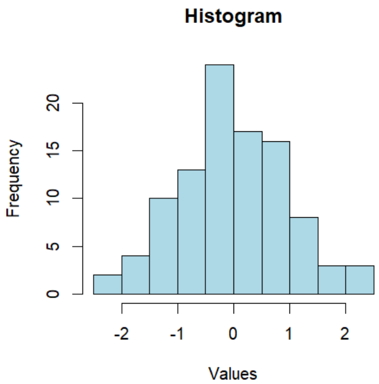
```r
#Question 1

#Here Random variable X = No. of times getting 6
# P(a <= X <= b) = P(X <= b) - P(X < a) = P(X <= b) - P(X < (a-1))
# P(7 <= X <= 9) = P(X <= 9) - P(X <= 6)
Desired_Prob <- pbinom(9, size = 12, prob = (1/6), lower.tail = TRUE) - pbinom(6, size = 12,
prob = (1/6), lower.tail = TRUE)
cat("Desired Probability: ", Desired_Prob, "\n")




#Question 2
#Here the Random variable X --> Scoring Marks
#We have to find P(X >= 84)
# P(X >= 84) = 1 - P(X <= 84)
# Bingo we can calculate P(X <= 84) as it is the CDF. Oh my God!

pX <- 1- pnorm(84, mean = 72, sd = 15.2, lower.tail = TRUE)
cat("Percentage of Students scoring 84: ", pX * 100, "\n")



#Question 3

#Here Random variable X --> No. of Cars arriving from 10AM to 11PM, lambda = 5 (car wash
every hour)
#Probability that no car arrives during this time
 dpois(0, lambda = 5) #Here we are using dpois because we want to find P(X = 0)

 #Random variable Y --> No. of customers that appers from 8 AM to 6PM, here lamba will be 5 *
10
 #We have to find P(48<=X<=50) = P(X<=50) - P(X<=47)
 pY <- ppois(50, lambda = 50, lower.tail = TRUE) - ppois(47, lambda = 50, lower.tail = TRUE)
 #Note: Here we have used ppois as we are finding P(X <= something)
 cat("Probability: ", pY, "\n")


#Question 4

N <- 250 #Total Processors (Population)
K <- 17  #Total no. of defective in Population
n <- 5   #Sample Size
k <- 3   #Total defective in sample

prob <- dhyper(k , K, N-K, n)
cat("Probability of exactly 3 defective processors: ", prob, "\n")
```

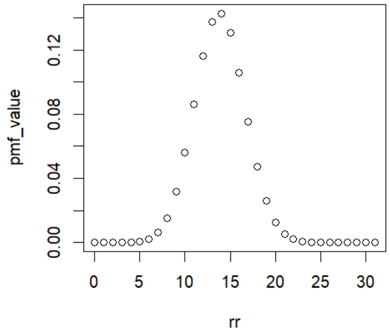
```r
#Question 5
#Part(a) --> Binomially Distributed

#Part(b)
bino_pmf <- function(r, n, p) {
  prob_X_equals_r <- choose(n, r) * p^r * (1-p)^(n-r)
  return(prob_X_equals_r)
}

#Given values
rr <- seq(0, 31, 1) #Sequence of possible X (0 to 31)
n <- 31
p <- 0.447
pmf_value <- numeric() #Declaring a empty numeric vector

for(i in 1:length(rr)) {
  pmf_value[i] <- bino_pmf(rr[i], n, p)
}

plot(rr, pmf_value)


#Part(C)
RR <- seq(0, 31, 1) #Sequence of possible X (0 to 31)
n <- 31
p <- 0.447
cdf_value <- numeric()

for(i in 1:length(RR)) {
  cdf_value[i] <- pbinom(RR[i], n, p)
}

plot(RR, cdf_value, col="red", pch = 19, lwd = 2)

#Part(D)
mean <- n*p
var <- n*p*(1-p)
sd <- sqrt(var)

cat("Mean: ", mean, "\n")
cat("Variance: ", var, "\n")
cat("Standard Deviation: ", sd, "\n")
```

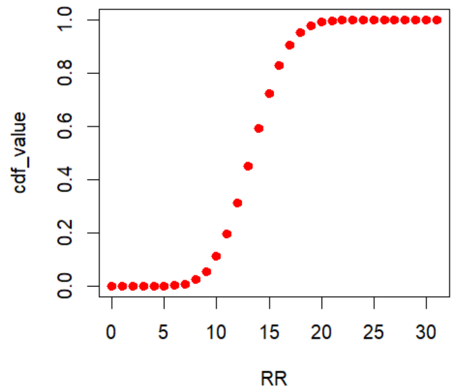
OUTPUT:

```
> #Here Random variable X = No. of times getting 6
> # P(a <= X <= b) = P(X <= b) - P(X < a) = P(X <= b) - P(X < (a-1))
> # P(7 <= X <= 9) = P(X <= 9) - P(X <= 6)
> Desired_Prob <- pbinom(9, size = 12, prob = (1/6), lower.tail = TRUE) - pbinom(6, size = 12, prob = (1/6), lower.tail = TRUE)
> cat("Desired Probability: ", Desired_Prob, "\n")
Desired Probability:  0.001291758
> pX <- 1- pnorm(84, mean = 72, sd = 15.2, lower.tail = TRUE)
> cat("Percentage of Students scoring 84: ", pX * 100, "\n")
Percentage of Students scoring 84:  21.49176
> #Here Random variable X --> No. of Cars arriving from 10AM to 11PM, lambda = 5 (car wash every hour)
> #Probability that no car arrives during this time
>  dpois(0, lambda = 5) #Here we are using dpois because we want to find P(X = 0)
[1] 0.006737947
>  #Random variable Y --> No. of customers that appers from 8 AM to 6PM, here lamba will be 5 * 10
>  #We have to find P(48<=X<=50) = P(X<=50) - P(X<=47)
>  pY <- ppois(50, lambda = 50, lower.tail = TRUE) - ppois(47, lambda = 50, lower.tail = TRUE)
>  #Note: Here we have used ppois as we are finding P(X <= something)
>  cat("Probability: ", pY, "\n")
Probability:  0.1678485
> N <- 250 #Total Processors (Population)
> K <- 17  #Total no. of defective in Population
> n <- 5   #Sample Size
> k <- 3   #Total defective in sample
> prob <- dhyper(k , K, N-K, n)
> cat("Probability of exactly 3 defective processors: ", prob, "\n")
Probability of exactly 3 defective processors:  0.002351153
> #Part(b)
> bino_pmf <- function(r, n, p) {
+   prob_X_equals_r <- choose(n, r) * p^r * (1-p)^(n-r)
+   return(prob_X_equals_r)
+ }
> #Given values
> rr <- seq(0, 31, 1) #Sequence of possible X (0 to 31)
> n <- 31
> p <- 0.447
> pmf_value <- numeric() #Declaring a empty numeric vector
> for(i in 1:length(rr)) {
+   pmf_value[i] <- bino_pmf(rr[i], n, p)
+ }
> plot(rr, pmf_value)
> #Part(C)
> RR <- seq(0, 31, 1) #Sequence of possible X (0 to 31)
```

```
> RR <- seq(0, 31, 1) #Sequence of possible X (0 to 31)
> n <- 31
> p <- 0.447
> cdf_value <- numeric()
> for(i in 1:length(RR)) {
+   cdf_value[i] <- pbinom(RR[i], n, p)
+ }
> plot(RR, cdf_value, col="red", pch = 19, lwd = 2)
> #Part(D)
> mean <- n*p
> var <- n*p*(1-p)
> sd <- sqrt(var)
> cat("Mean: ", mean, "\n")
Mean:  13.857
> cat("Variance: ", var, "\n")
Variance:  7.662921
> cat("Standard Deviation: ", sd, "\n")
Standard Deviation:  2.768198
>
```







Probability and Statistics (UMA401)
Experiment 4
(Mathematical Expectation, Moments and Functions of Random Variables)

```
#Question 1
x <- c(0, 1, 2, 3, 4)
p <- c(0.41, 0.37, 0.16, 0.05, 0.01)

#Expected value using Sum function
Exp_val <- sum(x * p)
cat("Expected value using Sum: ", Exp_val, "\n")

#Expected value using weighted.mean()
Exp_val <- weighted.mean(x, p)
cat("Expected value using Weighted Mean: ", Exp_val, "\n")

#Expected value using Matrix Multiplication
Exp_val <- c(x %*% p)
cat("Expected value using Matrix Multiplication: ", Exp_val, "\n")



#Question 2
#For Expected value of Continuous distribution --> integrate(x f(x)dx) from -Inf to Inf

#Defining the function for Integration
func <- function(t) {
  t * (0.1 * exp(-0.1 * t))
}

#Integrating from 0 to Inf (Not from -Inf to Inf as below 0 the f(t) = 0)
#Here $value is used to remove error from the answer
Exp_val_T <- integrate(func, lower = 0, upper = Inf)$value
```

```r
print(Exp_val_T)
cat("Expected value of T: ", Exp_val_T, "\n")



#Question 3
#Total books purchased --> 3
#Total cost price --> 3 * 6 --> 18
#Let say we have sold X no. of copies then we gain 12*X rupees
#Remaining 3-X will be returned and we will gain (3-X)*2 rupees
# Y = Selling price - Cost price
# Y = h(x) = 12*X + 2*(3-X) - 18 = 10X-12
x <- c(0, 1, 2, 3)
probX <- c(0.1, 0.2, 0.2, 0.5)
y <- 10*x-12
probY <- probX

Exp_val_Y <- sum(y*probY)
cat("Expected value of Y: ", Exp_val_Y, "\n")




#Question 4
func_first_moment <- function(x) {
  x * 0.5 * exp(-abs(x))
}

func_second_moment <- function(x) {
  x^2 * 0.5 * exp(-abs(x))
}

Exp_X <- integrate(func_first_moment, lower = 1, upper = 10)$value
Exp_X2 <- integrate(func_second_moment, lower = 1, upper = 10)$value

mean <- Exp_X
var <- Exp_X2 - Exp_X^2

cat("Mean: ", mean, "\n")
cat("Variance: ", var, "\n")



#Question 5
func_y <- function(y) {
  (3/4) * (1/4)^(sqrt(y)-1)
}
x <- 3
y <- x^2

prob_y <- func_y(y)
```

```
cat("Probablity of Y for X=3: ", prob_y, "\n")

#Declaring values
x <- c(1, 2, 3, 4, 5)
y <- x^2
prob_y <- func_y(y)
#Expected value
Exp_Y <- sum(y * prob_y)
cat("Expected Value: ", Exp_Y, "\n")

#Variance
Exp_Y2 <- sum(y^2 * prob_y)

var <- Exp_Y2 - Exp_Y^2
cat("Variance: ", var, "\n")
```

OUTPUT:

```
> #Question 1
> x <- c(0, 1, 2, 3, 4)
> p <- c(0.41, 0.37, 0.16, 0.05, 0.01)
> #Expected value using Sum function
> Exp_val <- sum(x * p)
> cat("Expected value using Sum: ", Exp_val, "\n")
Expected value using Sum:  0.88
> #Expected value using weighted.mean()
> Exp_val <- weighted.mean(x, p)
> cat("Expected value using Weighted Mean: ", Exp_val, "\n")
Expected value using Weighted Mean:  0.88
> #Expected value using Matrix Multiplication
> Exp_val <- c(x %*% p)
> cat("Expected value using Matrix Multiplication: ", Exp_val, "\n")
Expected value using Matrix Multiplication:  0.88
> #Defining the function for Integration
> func <- function(t) {
+   t * (0.1 * exp(-0.1 * t))
+ }
> #Integrating from 0 to Inf (Not from -Inf to Inf as below 0 the f(t) = 0)
> #Here $value is used to remove error from the answer
> Exp_val_T <- integrate(func, lower = 0, upper = Inf)$value
> print(Exp_val_T)
[1] 10
> cat("Expected value of T: ", Exp_val_T, "\n")
Expected value of T:  10
> #Question 3
> #Total books purchased --> 3
> #Total cost price --> 3 * 6 --> 18
> #Let say we have sold X no. of copies then we gain 12*X rupees
> #Remaining 3-X will be returned and we will gain (3-X)*2 rupees
> # Y = Selling price - Cost price
> # Y = h(x) = 12*X + 2*(3-X) - 18 = 10X-12
> x <- c(0, 1, 2, 3)
> probX <- c(0.1, 0.2, 0.2, 0.5)
> y <- 10*x-12
> probY <- probX
> Exp_val_Y <- sum(y*probY)
> cat("Expected value of Y: ", Exp_val_Y, "\n")
Expected value of Y:  9
```

```
> #Question 4
> func_first_moment <- function(x) {
+   x * 0.5 * exp(-abs(x))
+ }
> func_second_moment <- function(x) {
+   x^2 * 0.5 * exp(-abs(x))
+ }
> Exp_X <- integrate(func_first_moment, lower = 1, upper = 10)$value
> Exp_X2 <- integrate(func_second_moment, lower = 1, upper = 10)$value
> mean <- Exp_X
> var <- Exp_X2 - Exp_X^2
> cat("Mean: ", mean, "\n")
Mean:  0.3676297
> cat("Variance: ", var, "\n")
Variance:  0.7817776
> #Question 5
> func_y <- function(y) {
+   (3/4) * (1/4)^(sqrt(y)-1)
+ }
> x <- 3
> y <- x^2
> prob_y <- func_y(y)
> cat("Probablity of Y for X=3: ", prob_y, "\n")
Probablity of Y for X=3:  0.046875
> #Declaring values
> x <- c(1, 2, 3, 4, 5)
> y <- x^2
> prob_y <- func_y(y)
> #Expected value
> Exp_Y <- sum(y * prob_y)
> cat("Expected Value: ", Exp_Y, "\n")
Expected Value:  2.182617
> #Variance
> Exp_Y2 <- sum(y^2 * prob_y)
> var <- Exp_Y2 - Exp_Y^2
> cat("Variance: ", var, "\n")
Variance:  7.614112
> 
```

Probability and Statistics (UMA401)
Experiment 5
(Continuous Probability Distributions)

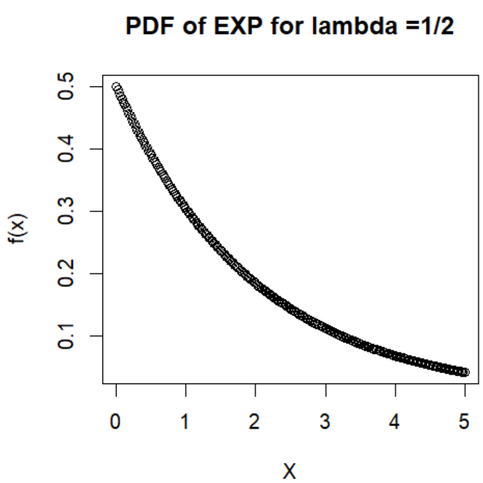
```
#Question 1
#(a)
#P(X>45)=1-P(X<=45)=1-F(45)
P_i<-1-punif(45,min=0,max=60)
print(P_i)

#OR

P_i<-punif(45,min=0,max=60,lower.tail = F)
print(P_i)

#(b) P(20<X<30) = P(X<30) - p(X<=20) = P(X<=30)-p(X<=20) = F(30)-F(20)
P_ii <- punif(30, min=0, max = 60) - punif(20, min = 0, max = 60)
print(P_ii)

#Question 2
#Here Exponential distribution is used --> It is used for reliability problems

#(a)
Exp_pdf <- function(x, lambda) {
  f_X <- lambda * exp(-lambda*x)
  return(f_X)
}

a <- as.integer(readline(prompt = "Enter the value of x: "))
P_X_3 <- Exp_pdf(a, 1/2)
print(P_X_3)


#Or
P_X_3 <- dexp(3, rate = 1/2)
print(P_X_3)


#(b)
X<-seq(0,5,by=0.02)
f_X<-Exp_pdf(X,1/2)
print(f_X)
plot(X,f_X,xlab="X",ylab="f(x)",main="PDF of EXP for lambda =1/2")

#(c)
P_atmost_3 <- pexp(3, rate = 1/2)
print(P_atmost_3)

#(d)
X <- seq(0, 5, by = 0.05)
```

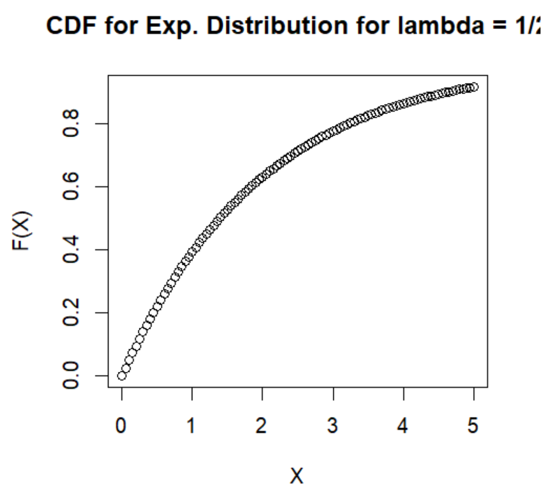
```
F_X <- pexp(X, rate = 1/2)
print(F_X)
plot(X, F_X, xlab = "X", ylab = "F(X)", main = "CDF for Exp. Distribution for lambda = 1/2")

#(e)
n <- 1000 #Because we want n to be large
X_sim <- rexp(n, rate = 1/2)
plot(density(X_sim), xlab = "Sim X", ylab = "Density", main = "Simulated Data")
hist(X_sim, probability = TRUE, xlab = "Sim X", ylab = "Density", main = "Simulated Data")
plot(X,f_X,xlab="X",ylab="f(x)",main="PDF of EXP for lambda =1/2")

plot(X,f_X,xlab="X",ylab="f(x)",main="PDF of EXP for lambda =1/2")


#Question 3 P(X>=1) = 1-P(X<1) = 1-F(1)
#Here generalized gamma distribution is used
alpha <- 2
beta <- 1/3
P_i <- pgamma(1, shape = alpha, scale = beta, lower.tail = F)
print(P_i)

#or
P_I <- 1- pgamma(1, shape = alpha, scale = beta, lower.tail = T)
print(P_I)


#(b)
prob <- 0.70
q_i <- qgamma(0.70, shape = alpha, scale = beta)
print(q_i)
```

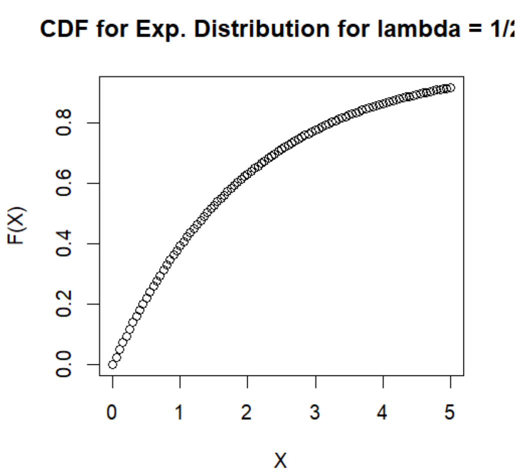
OUTPUT:

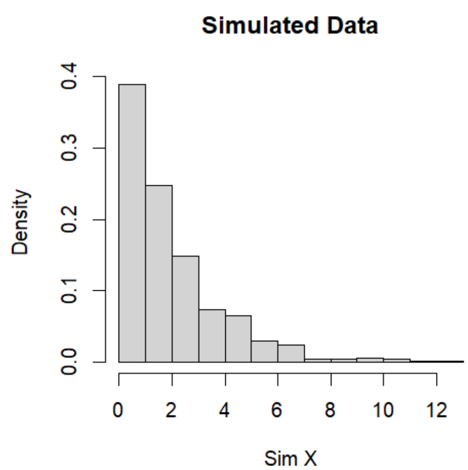
```
> #Question 1
> #(a)
> #P(X>45)=1-P(X<=45)=1-F(45)
> P_i<-1-punif(45,min=0,max=60)
> print(P_i)
[1] 0.25
> P_i<-punif(45,min=0,max=60,lower.tail = F)
> print(P_i)
[1] 0.25
> #(b) P(20<X<30) = P(X<30) - p(X<=20) = P(X<=30)-p(X<=20) = F(30)-F(20)
> P_ii <- punif(30, min=0, max = 60) - punif(20, min = 0, max = 60)
> print(P_ii)
[1] 0.1666667
> #(a)
> Exp_pdf <- function(x, lambda) {
+   f_X <- lambda * exp(-lambda*x)
+   return(f_X)
+ }
> a <- as.integer(readline(prompt = "Enter the value of x: "))
Enter the value of x: 3
> P_X_3 <- Exp_pdf(a, 1/2)
> print(P_X_3)
[1] 0.1115651
> #Or
> P_X_3 <- dexp(3, rate = 1/2)
> print(P_X_3)
[1] 0.1115651
> #(b)
> X<-seq(0,5,by=0.02)
> f_X<-Exp_pdf(X,1/2)
> print(f_X)
  [1] 0.50000000 0.49502492 0.49009934 0.48522277 0.48039472 0.47561471 0.47088227 0.46619691 0.46155817 0.45696559 0.45241871
 [12] 0.44791707 0.44346022 0.43904772 0.43467912 0.43035399 0.42607189 0.42183241 0.41763511 0.41347957 0.40936538 0.40529212
 [23] 0.40125940 0.39726680 0.39331393 0.38940039 0.38552579 0.38168975 0.37789187 0.37413178 0.37040911 0.36672348 0.36307452
 [34] 0.35946187 0.35588516 0.35234404 0.34883816 0.34536717 0.34193070 0.33852844 0.33516002 0.33182513 0.32852341 0.32525455
 [45] 0.32201821 0.31881408 0.31564182 0.31250113 0.30939170 0.30631320 0.30326533 0.30024779 0.29726027 0.29430248 0.29137413
 [56] 0.28847491 0.28560453 0.28276272 0.27994918 0.27716364 0.27440582 0.27167543 0.26897222 0.26629590 0.26364621 0.26102289
 [67] 0.25842567 0.25585429 0.25330850 0.25078803 0.24829265 0.24582210 0.24337613 0.24095450 0.23855696 0.23618328 0.23383321
 [78] 0.23150653 0.22920301 0.22692240 0.22466448 0.22242903 0.22021583 0.21802464 0.21585526 0.21370747 0.21158104 0.20947577
 [89] 0.20739146 0.20532788 0.20328483 0.20126211 0.19925952 0.19727686 0.19531392 0.19337051 0.19144644 0.18954152 0.18765555
[100] 0.18578835 0.18393972 0.18210949 0.18029747 0.17850348 0.17672734 0.17496887 0.17322791 0.17150426 0.16979776 0.16810825
[111] 0.16643554 0.16477948 0.16313990 0.16151663 0.15990951 0.15831838 0.15674309 0.15518347 0.15363937 0.15211063 0.15059711
[122] 0.14909864 0.14761508 0.14614629 0.14469211 0.14325240 0.14182701 0.14041581 0.13901865 0.13763539 0.13626590 0.13491003
[133] 0.13356765 0.13223863 0.13092283 0.12962013 0.12833039 0.12705348 0.12578928 0.12453765 0.12329848 0.12207164 0.12085701
[144] 0.11965446 0.11846388 0.11728514 0.11611814 0.11496274 0.11381884 0.11268633 0.11156508 0.11045499 0.10935594 0.10826783
[155] 0.10719055 0.10612399 0.10506804 0.10402259 0.10298755 0.10196281 0.10094826 0.09994381 0.09894935 0.09796479 0.09699002
[166] 0.09602495 0.09506949 0.09412353 0.09318699 0.09225976 0.09134176 0.09043290 0.08953307 0.08864220 0.08776020 0.08688697
[177] 0.08602243 0.08516649 0.08431907 0.08348008 0.08264944 0.08182707 0.08101288 0.08020678 0.07940871 0.07861858 0.07783632
[188] 0.07706183 0.07629505 0.07553590 0.07478431 0.07404019 0.07330348 0.07257410 0.07185197 0.07113704 0.07042921 0.06972843
[199] 0.06903462 0.06834771 0.06766764 0.06699434 0.06632773 0.06566776 0.06501436 0.06436745 0.06372698 0.06309289 0.06246511
[210] 0.06184357 0.06122821 0.06061898 0.06001581 0.05941865 0.05882742 0.05824208 0.05766256 0.05708881 0.05652077 0.05595837
[221] 0.05540158 0.05485032 0.05430455 0.05376422 0.05322925 0.05269961 0.05217524 0.05165609 0.05114210 0.05063323 0.05012942
[232] 0.04963063 0.04913679 0.04864787 0.04816382 0.04768458 0.04721011 0.04674036 0.04627529 0.04581484 0.04535898 0.04490765
[243] 0.04446081 0.04401842 0.04358043 0.04314679 0.04271748 0.04229243 0.04187161 0.04145498 0.04104250
> plot(X,f_X,xlab="X",ylab="f(x)",main="PDF of EXP for lambda =1/2")
> #(c)
> P_atmost_3 <- pexp(3, rate = 1/2)
> print(P_atmost_3)
[1] 0.7768698
> #(d)
> X <- seq(0, 5, by = 0.05)
> F_X <- pexp(X, rate = 1/2)
> print(F_X)
  [1] 0.00000000 0.02469009 0.04877058 0.07225651 0.09516258 0.11750310 0.13929202 0.16054298 0.18126925 0.20148378 0.22119922
 [12] 0.24042788 0.25918178 0.27747265 0.29531191 0.31271072 0.32967995 0.34623021 0.36237185 0.37811494 0.39346934 0.40844464
 [23] 0.42305019 0.43729513 0.45118836 0.46473857 0.47795422 0.49084358 0.50341470 0.51567543 0.52763345 0.53929622 0.55067104
 [34] 0.56176501 0.57258507 0.58313798 0.59343034 0.60346858 0.61325898 0.62280765 0.63212056 0.64120353 0.65006225 0.65870224
 [45] 0.66712892 0.67534753 0.68336323 0.69118102 0.69880579 0.70624230 0.71349520 0.72056903 0.72746821 0.73419704 0.74075974
 [56] 0.74716040 0.75340304 0.75949154 0.76542971 0.77122127 0.77686984 0.78237894 0.78775203 0.79299245 0.79810348 0.80308832
 [67] 0.80795009 0.81269182 0.81731648 0.82182695 0.82622606 0.83051655 0.83470111 0.83878236 0.84276283 0.84664503 0.85043138
 [78] 0.85412424 0.85772593 0.86123869 0.86466472 0.86800616 0.87126510 0.87444357 0.87754357 0.88056703 0.88351584 0.88639185
 [89] 0.88919684 0.89193258 0.89460078 0.89720309 0.89974116 0.90221656 0.90463084 0.90698551 0.90928205 0.91152188 0.91370641
[100] 0.91583701 0.91791500
```

```
> plot(X, F_X, xlab = "X", ylab = "F(X)", main = "CDF for Exp. Distribution for lambda = 1/2")
> #(e)
> n <- 1000 #Because we want n to be large
> X_sim <- rexp(n, rate = 1/2)
> plot(density(X_sim), xlab = "Sim X", ylab = "Density", main = "Simulated Data")
> hist(X_sim, probability = TRUE, xlab = "Sim X", ylab = "Density", main = "Simulated Data")
> plot(X,f_X,xlab="X",ylab="f(x)",main="PDF of EXP for lambda =1/2")
Error in xy.coords(x, y, xlabel, ylabel, log) :
  'x' and 'y' lengths differ
> plot(X,f_X,xlab="X",ylab="f(x)",main="PDF of EXP for lambda =1/2")
Error in xy.coords(x, y, xlabel, ylabel, log) :
  'x' and 'y' lengths differ
> #Question 3 P(X>=1) = 1-P(X<1) = 1-F(1)
> #Here generalized gamma distribution is used
> alpha <- 2
> beta <- 1/3
> P_i <- pgamma(1, shape = alpha, scale = beta, lower.tail = F)
> print(P_i)
[1] 0.1991483
> #or
> P_I <- 1- pgamma(1, shape = alpha, scale = beta, lower.tail = T)
> print(P_I)
[1] 0.1991483
> #(b)
> prob <- 0.70
> q_i <- qgamma(0.70, shape = alpha, scale = beta)
> print(q_i)
[1] 0.8130722
>
```









Thapar Institute of Engineering & Technology, Patiala
Department of Mathematics
LAB Experiment 6: Joint probability mass and density functions

```
#Question 1

# Define function f(x, y)
f <- function(x, y) {
  (x + y) / 30
}

# Define x and y ranges
x <- c(0:3)
y <- c(0:2)

#(i) Create matrix M1 using f(x, y)
M1 <- matrix(c(f(0, 0:2), f(1, 0:2), f(2, 0:2), f(3, 0:2)), nrow=4, ncol=3, byrow=T)
print(M1)

#(ii) Check whether it is a joint probability mass function (jpmf)
sum_M1 <- sum(M1)
print(sum_M1)  # Should be close to 1 if it's a jpmf

#(iii) Find the marginal distribution g(x) for x = 0, 1, 2, 3
f_X <- apply(M1, 1, sum)
print(f_X)

#(iv) Find the marginal distribution h(y) for y = 0, 1, 2
f_Y <- apply(M1, 2, sum)
print(f_Y)

#(v) Find the conditional probability at x = 0 given y = 1
p_x0_y1 <- M1[1, 2] / f_Y[2]
print(p_x0_y1)

#(vi) Calculate expectations, variances, and covariances
E_X <- sum(x * f_X)
E_Y <- sum(y * f_Y)
E_X2 <- sum(x * x * f_X)
E_Y2 <- sum(y * y * f_Y)
Var_X <- E_X2 - E_X^2
Var_Y <- E_Y2 - E_Y^2

# Print the variances
print(Var_X)
print(Var_Y)

#Cov(X, Y) = E(X*Y) - E(x)*E(Y)
x <- c(0:3)
```

```r
y <- c(0:2)

f1 <- function(x, y) {
  x*y*(x+y)/30
}

M2 <- matrix(c(f1(0, 0:2), f1(1, 0:2), f1(2, 0:2), f1(3, 0:2)), nrow=4, ncol=3, byrow=T)
M2

E_XY <- sum(M2)
E_XY
cov_XY <- E_XY - E_X * E_Y
cov_XY

# rho = cov(X,Y)/sqrt(Var_X)*sqrt(var_y)
rho <- cov_XY/sqrt(Var_X*Var_Y)
rho


# ---------------------------------------------------------------------------------------

#Question 2
#(i) f(x,y) = (2/5) *(2*x+3*y)
library(pracma)
f <- function(x, y) {
  (2*(2*x+3*y)/5)
}

I <- integral2(f, xmin = 0, xmax = 1, ymin=0, ymax=1)
print(I$Q)

#(ii) Find the marginal distribution g(x) at x = 1
f_X1 <- function(y) f(1,y)
g_X1 <- integral(f_X1, 0, 1)
g_X1

#(iii) Find the marginal distribution h(y) at y = 0
f_Y0 <- function(x) f(x, 0)
hy0 <- integral(f_Y0, 0, 1)
print(hy0)

#(iv) Find the expected value fo g(x, y) = xy
f_XY <- function(x, y) {x*y*f(x,y)}
E_XY <- integral2(f_XY,xmin = 0,xmax=1 ,ymin=0 ,ymax=1)
E_XY
print(E_XY$Q)
```

OUTPUT:

```
> # Define function f(x, y)
> f <- function(x, y) {
+   (x + y) / 30
+ }
> # Define x and y ranges
> x <- c(0:3)
> y <- c(0:2)
> #(i) Create matrix M1 using f(x, y)
> M1 <- matrix(c(f(0, 0:2), f(1, 0:2), f(2, 0:2), f(3, 0:2)), nrow=4, ncol=3, byrow=T)
> print(M1)
           [,1]       [,2]       [,3]
[1,] 0.00000000 0.03333333 0.06666667
[2,] 0.03333333 0.06666667 0.10000000
[3,] 0.06666667 0.10000000 0.13333333
[4,] 0.10000000 0.13333333 0.16666667
> #(ii) Check whether it is a joint probability mass function (jpmf)
> sum_M1 <- sum(M1)
> print(sum_M1)  # Should be close to 1 if it's a jpmf
[1] 1
> #(iii) Find the marginal distribution g(x) for x = 0, 1, 2, 3
> f_X <- apply(M1, 1, sum)
> print(f_X)
[1] 0.1 0.2 0.3 0.4
> #(iv) Find the marginal distribution h(y) for y = 0, 1, 2
> f_Y <- apply(M1, 2, sum)
> print(f_Y)
[1] 0.2000000 0.3333333 0.4666667
> #(v) Find the conditional probability at x = 0 given y = 1
> p_x0_y1 <- M1[1, 2] / f_Y[2]
> print(p_x0_y1)
[1] 0.1
> #(vi) Calculate expectations, variances, and covariances
> E_X <- sum(x * f_X)
> E_Y <- sum(y * f_Y)
> E_X2 <- sum(x * x * f_X)
> E_Y2 <- sum(y * y * f_Y)
> Var_X <- E_X2 - E_X^2
> Var_Y <- E_Y2 - E_Y^2
> # Print the variances
> print(Var_X)
[1] 1
> print(Var_Y)
> print(Var_Y)
[1] 0.5955556
> #Cov(X, Y) = E(X*Y) - E(x)*E(Y)
> x <- c(0:3)
> y <- c(0:2)
> f1 <- function(x, y) {
+   x*y*(x+y)/30
+ }
> M2 <- matrix(c(f1(0, 0:2), f1(1, 0:2), f1(2, 0:2), f1(3, 0:2)), nrow=4, ncol=3, byrow=T)
> M2
     [,1]       [,2]      [,3]
[1,]    0 0.00000000 0.0000000
[2,]    0 0.06666667 0.2000000
[3,]    0 0.20000000 0.5333333
[4,]    0 0.40000000 1.0000000
> E_XY <- sum(M2)
> E_XY
[1] 2.4
> cov_XY <- E_XY - E_X * E_Y
> cov_XY
[1] -0.1333333
> # rho = cov(X,Y)/sqrt(Var_X)*sqrt(var_y)
> rho <- cov_XY/sqrt(Var_X*Var_Y)
> rho
[1] -0.1727737
> #Question 2
> #(i) f(x,y) = (2/5) *(2*x+3*y)
> library(pracma)
> f <- function(x, y) {
+   (2*(2*x+3*y)/5)
+ }
> I <- integral2(f, xmin = 0, xmax = 1, ymin=0, ymax=1)
> print(I$Q)
[1] 1
> #(ii) Find the marginal distribution g(x) at x = 1
> f_X1 <- function(y) f(1,y)
> g_X1 <- integral(f_X1, 0, 1)
> g_X1
[1] 1.4
> #(iii) Find the marginal distribution h(y) at y = 0
> f_Y0 <- function(x) f(x, 0)
> hy0 <- integral(f_Y0, 0, 1)
```

```
> print(hy0)
[1] 0.4
> #(iv) Find the expected value fo g(x, y) = xy
> f_XY <- function(x, y) {x*y*f(x,y)}
> E_XY <- integral2(f_XY,xmin = 0,xmax=1 ,ymin=0 ,ymax=1)
> E_XY
$Q
[1] 0.3333333

$error
[1] 5.89806e-17

> print(E_XY$Q)
[1] 0.3333333
> 
```

Thapar Institute of Engineering & Technology, Patiala
Department of Mathematics
LAB Experiment 7: Central limit theorem

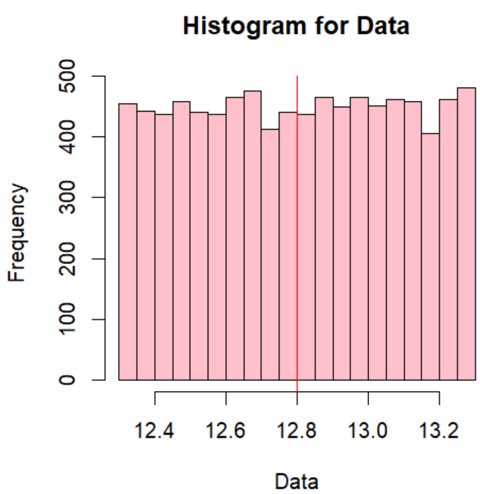
```r
#This assignment is based on --> Central Limit Theorem


# Part I
# (a) Import the data
data <- read.csv(file.choose())

# (b) Validate the data for correctness by counting the number of rows and viewing the top
then rows of the dataset.
dim(data)
#view top 10 row in data
head(data, 10)

# (c) Calculate the population mean and plot the observations by creating a histogram
mean(data$Wall.Thickness)

hist(data$Wall.Thickness, col = "pink", main = "Histogram for Data", xlab = "Data")

# (d)
abline(v=12.8, col = "red", lty=1)


# ---------------------------------------------------------------------------------------------
--------------

# Part II

# (a) Draw sufficient samples of size 10, calculate their means, and plot them in R using
# a histogram. Do you observe a normal distribution?
s10 <- c()
n<-9000

for (i in 1:n) {
  s10[i] = mean(sample(data$Wall.Thickness, 10, replace = T))
}

hist(s10, col = "lightgreen")
abline(v=mean(s10), col = "red")
abline(v=12.8, col = "blue")

# (b) Repeat the process with sample sizes of 50, 500, and 9000. Comment on your
observations.

s30 <- c()
s50 <- c()
s500 <- c()
```

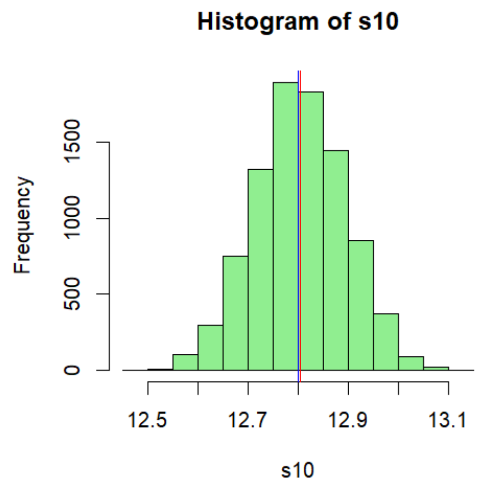
```r
n <- 9000

for(i in 1:n) {
  s30[i] <- mean(sample(data$Wall.Thickness, 30, replace = T))
  s50[i] <- mean(sample(data$Wall.Thickness, 50, replace = T))
  s500[i] <- mean(sample(data$Wall.Thickness, 500, replace = T))
}

par(mfrow = c(1, 3))
hist(s30, col = "red", main = "Sample of size 30")
abline(v=mean(s30), col = "blue")

hist(s50, col = "lightgreen", main = "Sample of size 30")
abline(v=mean(s50), col = "blue")

hist(s500, col = "orange", main = "Sample of size 30")
abline(v=mean(s500), col = "blue")
```

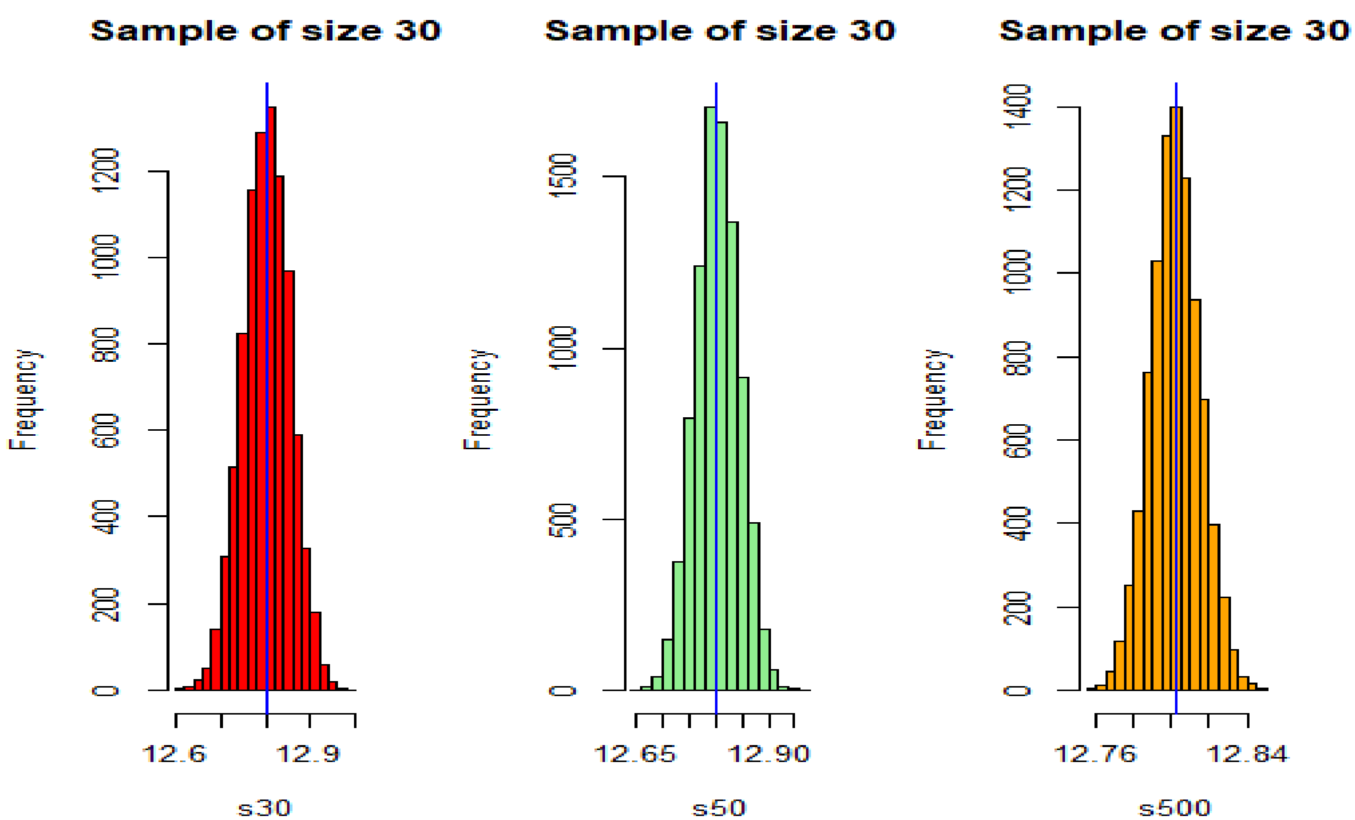
OUTPUT:

```
> # Part I
> # (a) Import the data
> data <- read.csv(file.choose())
> # (b) Validate the data for correctness by counting the number of rows and viewing the top then rows of the dataset.
> dim(data)
[1] 9000    1
> #view top 10 row in data
> head(data, 10)
   Wall.Thickness
1        12.35487
2        12.61742
3        12.36972
4        13.22335
5        13.15919
6        12.67549
7        12.36131
8        12.44468
9        12.62977
10       12.90381
> # (c) Calculate the population mean and plot the observations by creating a histogram
> mean(data$Wall.Thickness)
[1] 12.80205
> hist(data$Wall.Thickness, col = "pink", main = "Histogram for Data", xlab = "Data")
> # (d)
> abline(v=12.8, col = "red", lty=1)
> # (a) Draw sufficient samples of size 10, calculate their means, and plot them in R using
> # a histogram. Do you observe a normal distribution?
> s10 <- c()
> n<-9000
> for (i in 1:n) {
+   s10[i] = mean(sample(data$Wall.Thickness, 10, replace = T))
+ }
> hist(s10, col = "lightgreen")
> abline(v=mean(s10), col = "red")
> abline(v=12.8, col = "blue")
> s30 <- c()
> s50 <- c()
> s500 <- c()
> n <- 9000
> for(i in 1:n) {
+   s30[i] <- mean(sample(data$Wall.Thickness, 30, replace = T))
+   s50[i] <- mean(sample(data$Wall.Thickness, 50, replace = T))
```

```
+ }
> par(mfrow = c(1, 3))
> hist(s30, col = "red", main = "Sample of size 30")
> abline(v=mean(s30), col = "blue")
> hist(s50, col = "lightgreen", main = "Sample of size 30")
> abline(v=mean(s50), col = "blue")
> hist(s500, col = "orange", main = "Sample of size 30")
> abline(v=mean(s500), col = "blue")
>
```







LAB Experiment 8: Chi-square, t-distribution & F-distribution

```r
#Lab Assignment 8 (c)DanveerDagur
#Question 1
n<-100
df <- n-1
sample <- rt(n, df)
sample
hist(sample)

#Question 2
n <- 100
df <- c(2, 10, 25)
s1 <- rchisq(n, df[1])
s2 <- rchisq(n, df[2])
s3 <- rchisq(n, df[3])
mean(s1); var(s1)
mean(s2); var(s2)
mean(s3); var(s3)
par(mfrow = c(1, 3))
hist(s1)
hist(s2)
hist(s3)


#Question 3

x <- seq(-6, 6, length = 100)
df <- c(1, 4, 10, 30)
colour <- c("black", "red", "blue", "green")
#Find the value of t-dist
dt(x, df[1])
dt(x, df[2])
dt(x, df[3])
dt(x, df[4])
#Plot the density curve function
plot(x, dt(x, df[4]), type = "l", xlab = "t-value", ylab = "Density", col = colour[4])

for (i in 1:3) {
  lines(x, dt(x, df[i]), type = "l", xlab = "t-value", ylab = "Density", col = colour[i])
}

#Question 4

#Part(a)
#df = (10, 20 )
q95 <- qf(0.95, df1 = 10, df2 = 20)
q95

#Part(b)
```

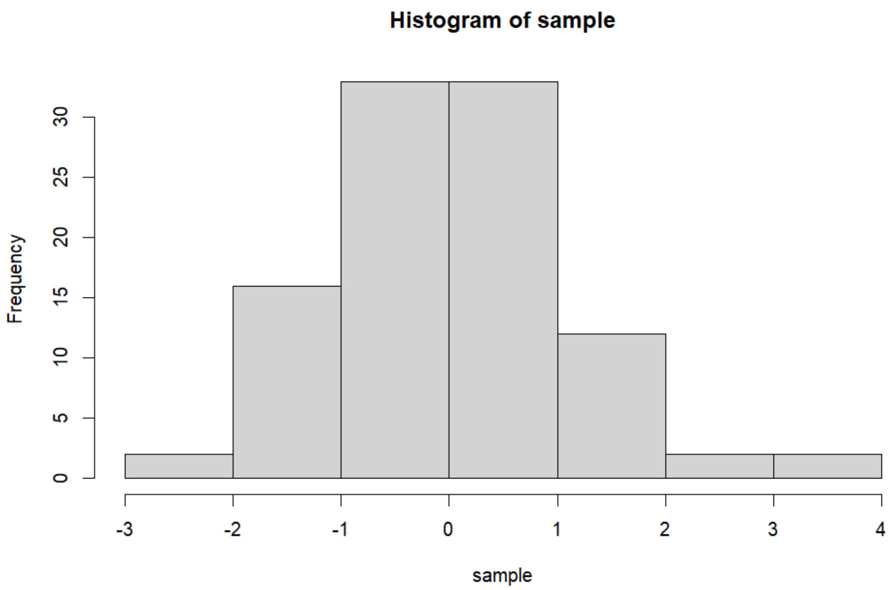
```
x <- 1.5
v1 <- 10
v2 <- 20
# int [0, 1.5]
pf(1.5, v1, v2, lower.tail = T)
# int [1.5, inf]
pf(1.5, v1, v2, lower.tail = F)

#Part(c)

q <- c(0.25, 0.5, 0.75, 0.999)
v1 <- 10
v2 <- 20
qf(q[1], v1, v2, lower.tail = T)
qf(q[2], v1, v2, lower.tail = T)
qf(q[3], v1, v2, lower.tail = T)
qf(q[4], v1, v2, lower.tail = T)
# n = 1000 plot hist
x <- rf(1000, df1 = v1, df2 = v2)
hist(x, df1 = v1, df2 = v2)
hist(x, breaks = 'scott', freq = FALSE, xlim = c(0, 3), ylim = c(0, 1)) #ye faltu ki line
hai, jyada dhyaan dhyaan mat do
```

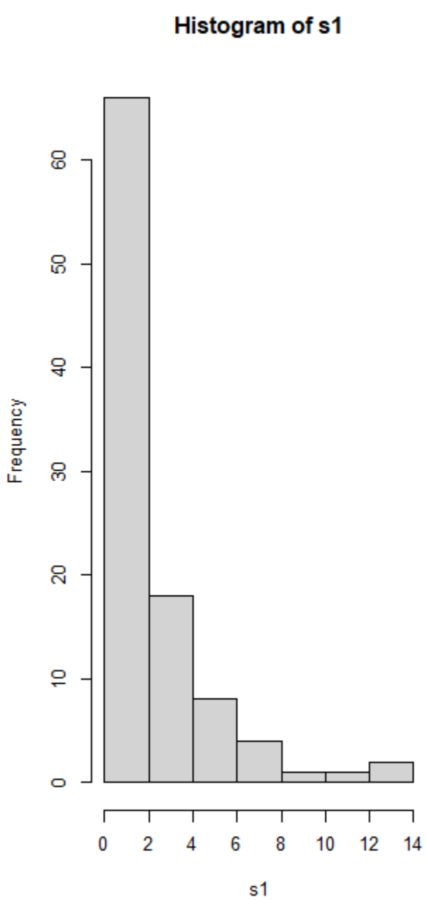
OUTPUT

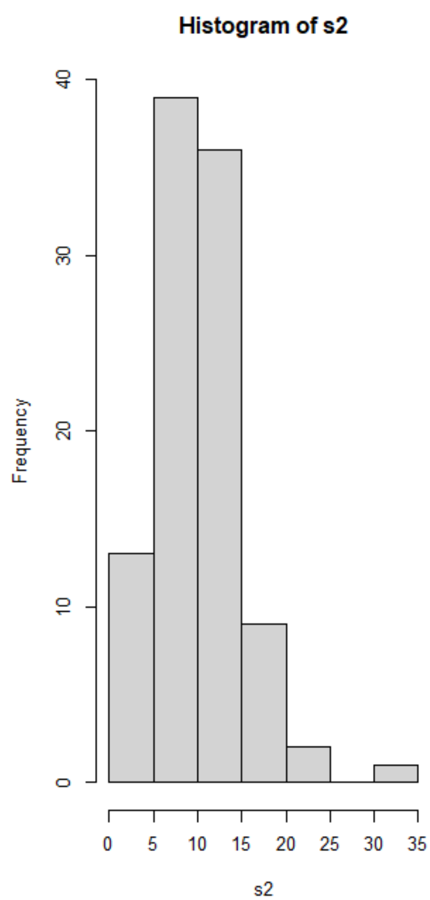
```
> #Lab Assignment 8 (c)DanveerDagur
> #Question 1
> n<-100
> df <- n-1
> sample <- rt(n, df)
> sample
  [1] -0.652434184 -0.273182501 -2.231594706  0.752087747  0.686851893  0.119346326 -0.473203373 -0.960959438 -0.379214328
 [10] -0.360674868  0.380624402  1.144790863 -0.852629233  0.390291908 -1.310774750 -1.317865819  0.072221765  0.211839491
 [19] -0.393097225 -0.007030018 -0.819806297 -1.475144710  1.345882967 -1.010894656 -1.967223951  0.504023061  0.513238180
 [28] -0.179728800  0.166960373  0.887450528 -0.808319657 -0.207924264  1.472668508  0.100090121  0.398230275  1.093305355
 [37] -0.051561359  2.077494358  1.729690742  0.735681765 -1.481499005 -0.560965061 -0.625321438 -0.461936451  1.025258662
 [46]  1.615721855  1.322608308 -0.227946834 -0.009763687  0.104518876 -1.262812827 -0.136048697 -0.352931885  1.914063732
 [55]  1.435723498 -1.257015324  0.310632238 -0.507287401  0.361248159  0.879797376 -1.999091785 -1.176801661 -0.756575526
 [64] -1.508200858  1.254946772 -0.434406529 -1.069289147 -0.941088306 -2.048156760 -1.568198122  0.400593459 -1.428757058
 [73] -1.095798878  0.846494927  0.068241282  0.241110178 -0.716547063 -0.476696715  0.156910813 -0.691120171  0.691475040
 [82]  0.767259195  0.554116299 -1.323570820 -0.936190872  0.954150728 -0.038979895  1.701703181  0.115649159  2.040027874
 [91]  0.164638911  3.384424348  0.151517702  0.513552034 -0.504409639  0.133802507  3.036268020 -0.156524401 -0.340633329
[100]  0.892240550
> hist(sample)
> #Question 2
> n <- 100
> df <- c(2, 10, 25)
> s1 <- rchisq(n, df[1])
> s2 <- rchisq(n, df[2])
> s3 <- rchisq(n, df[3])
> mean(s1); var(s1)
[1] 2.074526
[1] 6.353445
> mean(s2); var(s2)
[1] 10.15281
[1] 21.92758
> mean(s3); var(s3)
[1] 24.85268
[1] 37.24983
> par(mfrow = c(1, 3))
> hist(s1)
> hist(s2)
> hist(s3)
> x <- seq(-6, 6, length = 100)
> df <- c(1, 4, 10, 30)
```

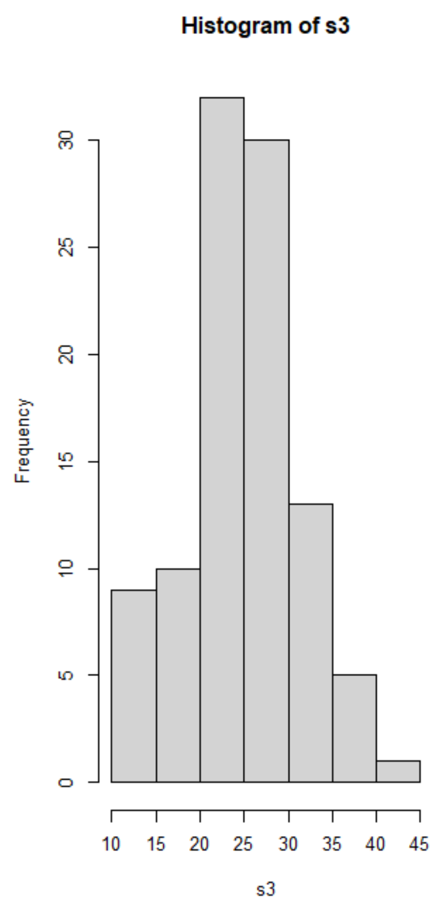
```
> df <- c(1, 4, 10, 30)
> colour <- c("black", "red", "blue", "green")
> #Find the value of t-dist
> dt(x, df[1])
  [1] 0.008602970 0.008951310 0.009321021 0.009713870 0.010131806 0.010576983 0.011051792 0.011558887 0.012101221 0.012682086
 [11] 0.013305165 0.013974580 0.014694962 0.015471523 0.016310143 0.017217477 0.018201075 0.019269524 0.020432624 0.021701588
 [21] 0.023089287 0.024610541 0.026282468 0.028124906 0.030160921 0.032417419 0.034925891 0.037723307 0.040853208 0.044367012
 [31] 0.048325591 0.052801137 0.057879356 0.063661977 0.070269505 0.077844030 0.086551677 0.096583858 0.108155840 0.121499988
 [41] 0.136849375 0.154405107 0.174278263 0.196396298 0.220368383 0.245321632 0.269758339 0.291538659 0.308123970 0.317144983
 [51] 0.317144983 0.308123970 0.291538659 0.269758339 0.245321632 0.220368383 0.196396298 0.174278263 0.154405107 0.136849375
 [61] 0.121499988 0.108155840 0.096583858 0.086551677 0.077844030 0.070269505 0.063661977 0.057879356 0.052801137 0.048325591
 [71] 0.044367012 0.040853208 0.037723307 0.034925891 0.032417419 0.030160921 0.028124906 0.026282468 0.024610541 0.023089287
 [81] 0.021701588 0.020432624 0.019269524 0.018201075 0.017217477 0.016310143 0.015471523 0.014694962 0.013974580 0.013305165
 [91] 0.012682086 0.012101221 0.011558887 0.011051792 0.010576983 0.010131806 0.009713870 0.009321021 0.008951310 0.008602970
> dt(x, df[2])
  [1] 0.001185854 0.001299674 0.001426572 0.001568291 0.001726840 0.001904535 0.002104055 0.002328498 0.002581463 0.002867130
 [11] 0.003190370 0.003556866 0.003973266 0.004447354 0.004988268 0.005606751 0.006315456 0.007129303 0.008065920 0.009146149
 [21] 0.010394664 0.011840692 0.013518866 0.015470216 0.017743327 0.020395643 0.023494940 0.027120922 0.031366892 0.036341391
 [31] 0.042169621 0.048994381 0.056976082 0.066291261 0.077128754 0.089682498 0.104139687 0.120662946 0.139365306 0.160277437
 [41] 0.183307807 0.208198657 0.234483644 0.261456453 0.288162552 0.313426933 0.335927310 0.354313737 0.367362749 0.374140500
 [51] 0.374140500 0.367362749 0.354313737 0.335927310 0.313426933 0.288162552 0.261456453 0.234483644 0.208198657 0.183307807
 [61] 0.160277437 0.139365306 0.120662946 0.104139687 0.089682498 0.077128754 0.066291261 0.056976082 0.048994381 0.042169621
 [71] 0.036341391 0.031366892 0.027120922 0.023494940 0.020395643 0.017743327 0.015470216 0.013518866 0.011840692 0.010394664
 [81] 0.009146149 0.008065920 0.007129303 0.006315456 0.005606751 0.004988268 0.004447354 0.003973266 0.003556866 0.003190370
 [91] 0.002867130 0.002581463 0.002328498 0.002104055 0.001904535 0.001726840 0.001568291 0.001426572 0.001299674 0.001185854
> dt(x, df[3])
  [1] 8.808511e-05 1.049214e-04 1.252258e-04 1.497602e-04 1.794627e-04 2.154911e-04 2.592754e-04 3.125844e-04 3.776092e-04
 [10] 4.570665e-04 5.543283e-04 6.735831e-04 8.200373e-04 1.000165e-03 1.222017e-03 1.495608e-03 1.833383e-03 2.250800e-03
 [19] 2.767036e-03 3.405837e-03 4.196543e-03 5.175295e-03 6.386451e-03 7.884205e-03 9.734397e-03 1.201647e-02 1.482550e-02
 [28] 1.827413e-02 2.249422e-02 2.763790e-02 3.387746e-02 4.140377e-02 5.042225e-02 6.114577e-02 7.378367e-02 8.852619e-02
 [37] 1.055239e-01 1.248621e-01 1.465323e-01 1.704005e-01 1.961789e-01 2.234026e-01 2.514189e-01 2.793936e-01 3.063382e-01
 [46] 3.311623e-01 3.527460e-01 3.700297e-01 3.821091e-01 3.883232e-01 3.883232e-01 3.821091e-01 3.700297e-01 3.527460e-01
 [55] 3.311623e-01 3.063382e-01 2.793936e-01 2.514189e-01 2.234026e-01 1.961789e-01 1.704005e-01 1.465323e-01 1.248621e-01
 [64] 1.055239e-01 8.852619e-02 7.378367e-02 6.114577e-02 5.042225e-02 4.140377e-02 3.387746e-02 2.763790e-02 2.249422e-02
 [73] 1.827413e-02 1.482550e-02 1.201647e-02 9.734397e-03 7.884205e-03 6.386451e-03 5.175295e-03 4.196543e-03 3.405837e-03
 [82] 2.767036e-03 2.250800e-03 1.833383e-03 1.495608e-03 1.222017e-03 1.000165e-03 8.200373e-04 6.735831e-04 5.543283e-04
 [91] 4.570665e-04 3.776092e-04 3.125844e-04 2.592754e-04 2.154911e-04 1.794627e-04 1.497602e-04 1.252258e-04 1.049214e-04
[100] 8.808511e-05
> dt(x, df[4])
  [1] 1.948678e-06 2.742971e-06 3.862943e-06 5.442161e-06 7.668593e-06 1.080643e-05 1.522639e-05 2.144773e-05 3.019610e-05
 [10] 4.248311e-05 5.971486e-05 8.383942e-05 1.175458e-04 1.645301e-04 2.298498e-04 3.203887e-04 4.454635e-04 6.176038e-04
 [19] 8.535416e-04 1.175449e-03 1.612457e-03 2.202481e-03 2.994355e-03 4.050262e-03 5.448382e-03 7.285618e-03 9.680204e-03
 [28] 1.277386e-02 1.673306e-02 2.174888e-02 2.803476e-02 3.582149e-02 4.534868e-02 5.685228e-02 7.054761e-02 8.660837e-02
 [37] 1.051419e-01 1.261628e-01 1.495662e-01 1.751045e-01 2.023705e-01 2.307906e-01 2.596315e-01 2.880217e-01 3.149896e-01
 [46] 3.395167e-01 3.606011e-01 3.773274e-01 3.889359e-01 3.948821e-01 3.948821e-01 3.889359e-01 3.773274e-01 3.606011e-01
 [55] 3.395167e-01 3.149896e-01 2.880217e-01 2.596315e-01 2.307906e-01 2.023705e-01 1.751045e-01 1.495662e-01 1.261628e-01
 [64] 1.051419e-01 8.660837e-02 7.054761e-02 5.685228e-02 4.534868e-02 3.582149e-02 2.803476e-02 2.174888e-02 1.673306e-02
 [73] 1.277386e-02 9.680204e-03 7.285618e-03 5.448382e-03 4.050262e-03 2.994355e-03 2.202481e-03 1.612457e-03 1.175449e-03
 [82] 8.535416e-04 6.176038e-04 4.454635e-04 3.203887e-04 2.298498e-04 1.645301e-04 1.175458e-04 8.383942e-05 5.971486e-05
 [91] 4.248311e-05 3.019610e-05 2.144773e-05 1.522639e-05 1.080643e-05 7.668593e-06 5.442161e-06 3.862943e-06 2.742971e-06
[100] 1.948678e-06
> #Plot the density curve function
> plot(x, dt(x, df[4]), type = "l", xlab = "t-value", ylab = "Density", col = colour[4])
> for (i in 1:3) {
+   lines(x, dt(x, df[i]), type = "l", xlab = "t-value", ylab = "Density", col = colour[i])
+ }
> #Part(a)
> #df = (10, 20 )
> q95 <- qf(0.95, df1 = 10, df2 = 20)
> q95
[1] 2.347878
> #Part(b)
> x <- 1.5
> v1 <- 10
> v2 <- 20
> # int [0, 1.5]
> pf(1.5, v1, v2, lower.tail = T)
[1] 0.7890535
> # int [1.5, inf]
> pf(1.5, v1, v2, lower.tail = F)
[1] 0.2109465
> q <- c(0.25, 0.5, 0.75, 0.999)
> v1 <- 10
> v2 <- 20
> qf(q[1], v1, v2, lower.tail = T)
[1] 0.6563936
> qf(q[2], v1, v2, lower.tail = T)
[1] 0.9662639
> qf(q[3], v1, v2, lower.tail = T)
[1] 1.399487
```

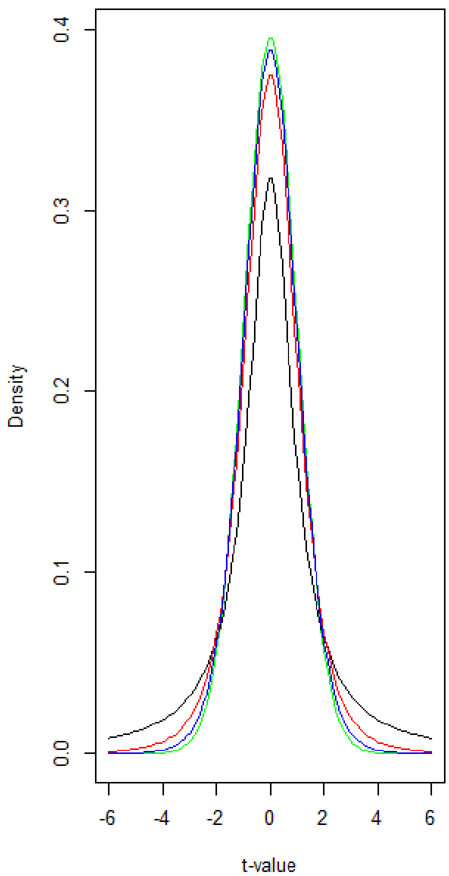
```
> qf(q[4], v1, v2, lower.tail = T)
[1] 5.075246
> # n = 1000 plot hist
> x <- rf(1000, df1 = v1, df2 = v2)
> hist(x, df1 = v1, df2 = v2)
```

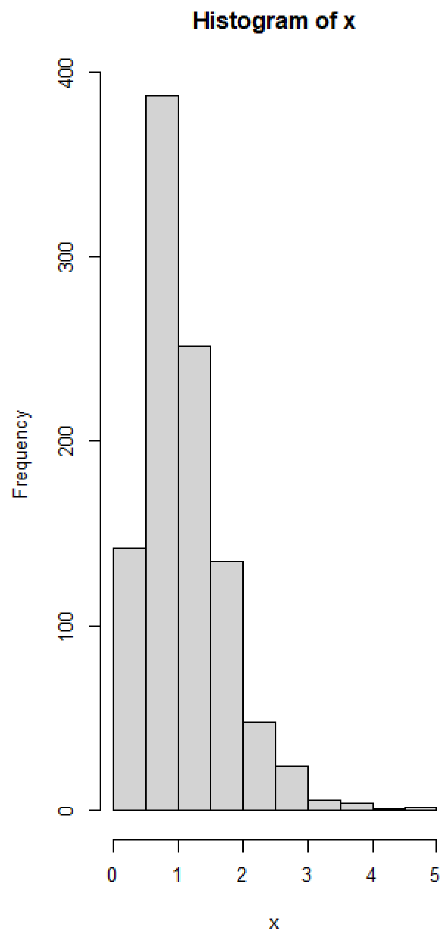

Histogram of sample
Frequency
0 5 10 15 20 25 30
-3 -2 -1 0 1 2 3 4
sample

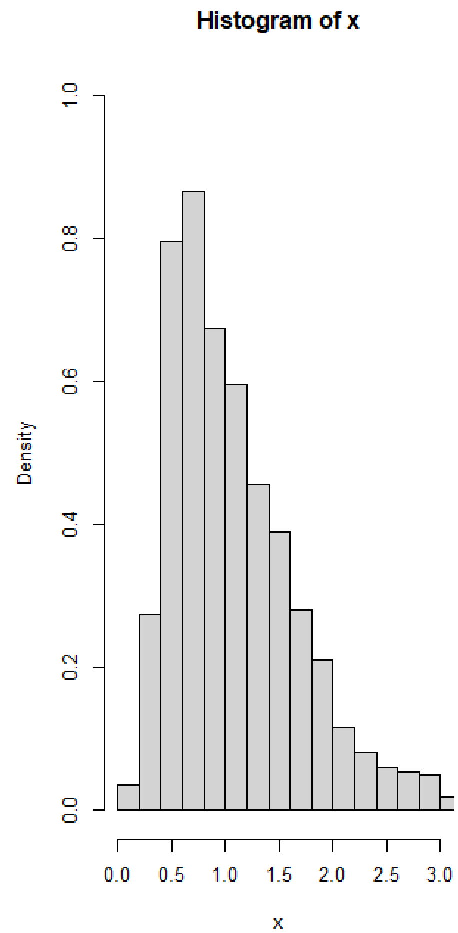

Histogram of s1
Frequency
0 10 20 30 40 50 60
0 2 4 6 8 10 12 14
s1


Histogram of s2
Frequency
0 10 20 30 40
0 5 10 15 20 25 30 35
s2


Histogram of s3
Frequency
0 5 10 15 20 25 30
10 15 20 25 30 35 40 45
s3


Density
t-value
0.0
0.1
0.2
0.3
0.4
-6
-4
-2
0
2
4
6


Histogram of x
Frequency
x
0
100
200
300
400
0
1
2
3
4
5


Histogram of x
Density
x
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0